Licenciatura em Engenharia Informática e de Computadores

Língua Natural

EXERCÍCIOS RESOLVIDOS

Nuno Mamede

3ª Edição
Setembro 2006

Departamento de Engenharia Informática
Instituto Superior Técnico

ÍNDICE

ENUNCIADOS 5

SOLUÇÕES 17

Problema 1 — Correcção sintáctica, semântica e pragmática 17

Problema 2 — Estruturas sintácticas de "As crianças foram encontradas mortas pela mãe" 17

Problema 3 — Estruturas sintácticas de "O bar necessita um homem para lavar pratos e duas empregadas" 18

Problema 4 — Estruturas sintácticas de "As crianças comeram o bolo com a colher" 19

Problema 5 — Estruturas sintácticas de "Atirei o homem em o lixo" 20

Problema 6 — Estruturas sintácticas de "Vi o homem no monte com o telescópio" 21

Problema 7 — Conversão de "potato" em "batata" com transdutores 23

Problema 8 — Dicionário multilingue com transdutores 25

Problema 9 — Distância de Levenshtein entre as palavras "bola" e "rolha" 28

Problema 10 — Distância mínima entre "intention" e "execution" 31

Problema 11 — Cálculo de bigramas num corpus com 300 frases 33

Problema 12 — Algoritmo de Viterbi em "O rato roeu a rolha" 34

Problema 13 — Análise descendente de "Os alunos dedicados passam" 38

Problema 14 — Análise descendente de "Os professores dão as aulas em as salas" 39

Problema 15 — Análise ascendente (grafo) de "A aluna de saia entrou pelas traseiras" . 40

Problema 16 — Análise ascendente (grafo) de "Senhora aluga quarto a rapariga em bom estado" 41

Problema 17 — Análise ascendente (grafo) de "Os professores dão as aulas em as salas" 44

Problema 18 — Análise descendente (grafo) de "A aluna de saia entrou pelas traseiras" 45

Problema 19 — Análise descendente (grafo) de "Senhora aluga quarto a rapariga em bom estado" 46

Problema 20 — Análise descendente (grafo) de "Os professores dão as aulas em as salas" 48

Problema 21 — Análise com algoritmo de Earley de "A aluna de saia entrou pelas traseiras" 49

Problema 22 — Análise com algoritmo de Earley "Senhora aluga quarto a rapariga em bom estado" 51

Problema 23 — Análise com algoritmo de Earley "Os professores dão as aulas em as salas" 52

Problema 24 — Análise (rede de transições) de "Um carro amarelo em o domingo" 55

Problema 25 — Análise (rede de transições) de "O Zé Maria ganhou o concurso BigBrother" 56

Problema 26 — Análise (rede de transições) de "Os professores dão as aulas em as salas" 57

Problema 27 — Frases aceites por uma rede de transições. 58

Problema 28 — Análise (rede de transições estendida) de "O prof viu o gato" 59

Problema 29 — Análise (rede de transições estendida) de "O pardal azul comeu a pevide que brilhava" 61

Problema 30 — Reconhecedor de números usando o Prolog (DCG) 64

Problema 31 — Análise de "Os professores dão as aulas em as salas" em Prolog (DCG) ... 65

Problema 32 — Verificação da sintaxe usando o Prolog (DCG) 66

Problema 33 — Verificação da sintaxe de 5 frases usando o Prolog (DCG) 66

Problema 34 — Conversão de uma frase para uma linguagem lógica com Prolog (DCG) . 67

Problema 35 — Representação na forma lógica e na forma quase-lógica 68

Problema 36 — Representação na forma quase-lógica usando funções temáticas 68

Problema 37 — Funções temáticas de sintagmas nominais 69

Problema 38 — Funções temáticas associadas a verbos 69

Problema 39 — Simplificação sintáctica de uma gramática com a característica SEM .. 70

Problema 40 — Cálculo da semântica de várias frases 70

Problema 41 — Rescrita de uma gramática usando as "convenções do Allen" 72

ENUNCIADOS

Problema 1 — Correcção sintáctica, semântica e pragmática

Classifique cada uma das seguintes frases, sabendo que a pessoa que a proferiu está a responder à queixa que o carro está muito frio. Use os seguintes vectores de classificação: (i) correcção sintáctica; (ii) correcção semântica e; (iii) correcção pragmática.

a) `O aquecimento estão ligado`
b) `Os pneus são novos`
c) `Muitas janelas estão a comer o guisado`

Problema 2 — Estruturas sintácticas de “As crianças foram encontradas mortas pela mãe”

Quais são as estruturas sintácticas que podem ser associadas à frase: “`As crianças foram encontradas mortas pela mãe`”. Qual ou quais são as interpretações semânticas que consegue associar a cada uma dessas estruturas sintácticas?
Considere o seguinte léxico e gramática:

Léxico:
```
a:           Det
as:          Det
crianças:    N
encontradas: V
foram:       Vaux
mãe:         N
mortas:      Adj,V
por:         P
```

Gramática:
```
F  -> SN SV
SV -> Vaux V Adj SP
SV -> Vaux V SV
SV -> V SP
SN -> Det N
SP -> P SN
```

Problema 3 — Estruturas sintácticas de “O bar necessita um homem para lavar pratos e duas empregadas”

Quais são as estruturas sintácticas que podem ser associadas à frase: “`O bar necessita um homem para lavar pratos e duas empregadas`”. Qual ou quais são as interpretações semânticas que consegue associar a cada uma dessas estruturas sintácticas?
Considere o seguinte léxico e gramática:

Léxico:
```
bar:        N
duas:       Card
e:          Conj
empregadas: N
homem:      N
lavar:      V
necessita:  V
o:          Det
para:       P
pratos:     N
um:         Card
```

Gramática:
```
F  -> SN SV
SV -> V SN
SN -> N
SN -> Det N
SN -> Card N
SN -> SN SP
SN -> SN Conj SN
SP -> P SV
```

Problema 4 — Estruturas sintácticas de "As crianças comeram o bolo com a colher"

Quais são as estruturas sintácticas que podem ser associadas à frase: "`As crianças comeram o bolo com a colher`". Qual ou quais são as interpretações semânticas que consegue associar a cada uma dessas estruturas sintácticas?
Considere o seguinte léxico e gramática:

Léxico:
```
a:        Det
as:       Det
bolo:     N
colher:   N
com:      P
comeram:  V
crianças: N
o:        Det
```

Gramática:
```
F  -> SN SV
SV -> V
SV -> V SN
SV -> SV SP
SN -> Det N
SN -> SN SP
SP -> P SN
```

Problema 5 — Estruturas sintácticas de "Atirei o homem em o lixo"

Quais são as estruturas sintácticas que podem ser associadas à frase: "`Atirei o homem em o lixo`". Qual ou quais são as interpretações semânticas que consegue associar a cada uma dessas estruturas sintácticas?

Léxico:
```
atirei:   V
em:       P
homem:    N
lixo:     N
o:        Det
```

Gramática:
```
F  -> SV
SV -> V
SV -> V SN
SV -> SV SP
SN -> Det N
SN -> SN SP
SP -> P SN
```

Problema 6 — Estruturas sintácticas de "Vi o homem no monte com o telescópio"

Quais são as estruturas sintácticas que podem ser associadas à frase: "`Vi o homem no monte com o telescópio`". Qual ou quais são as interpretações semânticas que consegue associar a cada uma dessas estruturas sintácticas?
Considere o seguinte léxico e gramática:

Léxico:
```
com:        P
em:         P
homem:      N
monte:      N
o:          Det
telescópio: N
vi:         V
```

Gramática:
```
F  -> SV
SV -> V
SV -> V SN
SV -> SV SP
SN -> Det N
SN -> SN SP
SP -> P SN
```

Problema 7 — Conversão de "potato" em "batata" com transdutores

Usando as ferramentas FSM Library e Graphviz, disponíveis no sítio da ATT, e considerando que o alfabeto de símbolos é constituído pelas letras do alfabeto, faça um pequeno tradutor que converte:

"`potato`" em "`batata`"
"`onion`" em "`cebola`
"`cabbage`" em "`couve`"

Problema 8 — Dicionário multilingue com transdutores

Usando as ferramentas FSM Library e Graphviz, disponíveis no sítio da ATT, faça um tradutor de nomes de animais entre qualquer uma das seguintes línguas: i) Português, ii) Francês, iii) Espanhol e iv) Inglês.

Problema 9 — Distância de Levenshtein entre as palavras "bola" e "rolha"

Calcule a distância de Levenshtein entre as palavras "`bola`" e "`rolha`".

Problema 10 — Distância mínima entre "intention" e "execution"

Use o algoritmo que calcula o número mínimo de edições entre cadeias de caracteres para calcular o número mínimo de operações necessárias à conversão da cadeia de caracteres "`intention`" na cadeia de caracteres "`execution`".
Considere que o custo de cada operação de inserção, remoção e substituição é 1 (distância de Levenshtein). Considere também que o custo de uma substituição é 2 e que o custo das outras operações é 1.

Problema 11 — Cálculo de bigramas num corpus com 300 frases

Considere um corpus com 300 frases compostas por 1998 palavras, das quais 833 são nomes, 300 são verbos, 558 são determinantes e 307 são preposições. Calcule os bigramas referentes aos pares para os quais se efectuaram as seguintes contagens:

Par	Contagem
ϕ,DET	213
ϕ,N	87
DET,N	558
N,V	358
N,N	108
N,P	366
V,N	75
V,DET	194
P,DET	226
P,N	81

Problema 12 — Algoritmo de Viterbi em "O rato roeu a rolha"

Use o algoritmo de Viterbi para proceder à análise morfológica da frase "`O rato roeu a rolha`", assumindo que existiu uma análise prévia de um corpus com 300 frases, o mesmo que foi usado no problema anterior (Problema 11). Considere também que se verificaram as seguintes contagens:

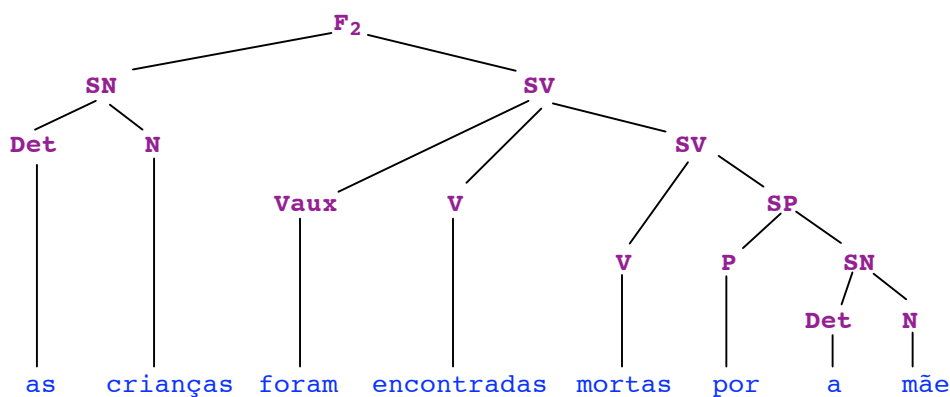
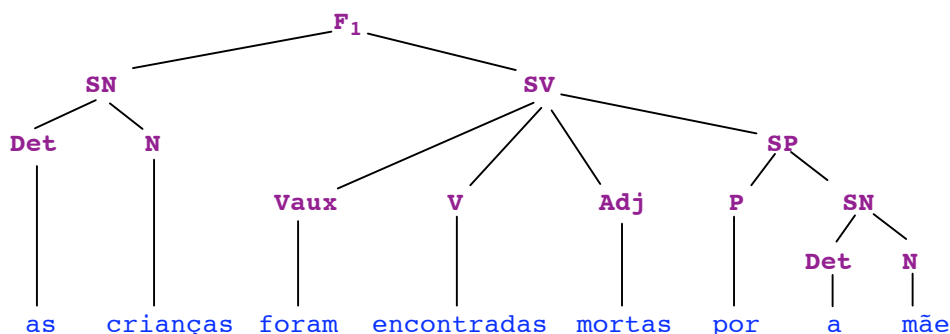
	N	V	DET	P	Total
o	0	0	250	87	337
rato	28	0	0	0	31
roeu	0	1	0	0	1
a	0	0	208	54	262
rolha	5	0	0	0	5
outras	810	299	100	166	1362
Total	833	300	558	307	1998

SOLUÇÕES

Problema 1 — Correcção sintáctica, semântica e pragmática

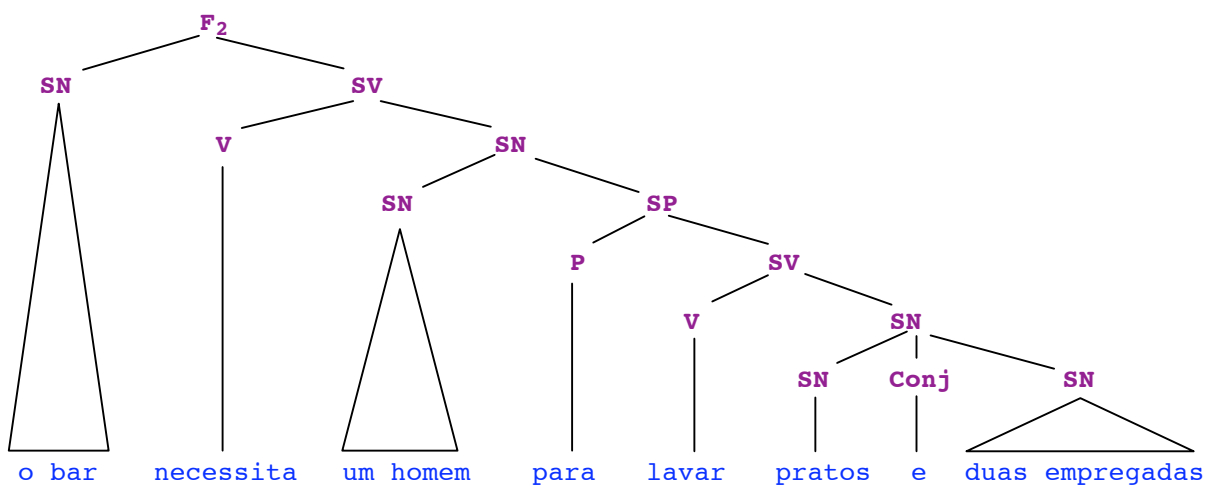
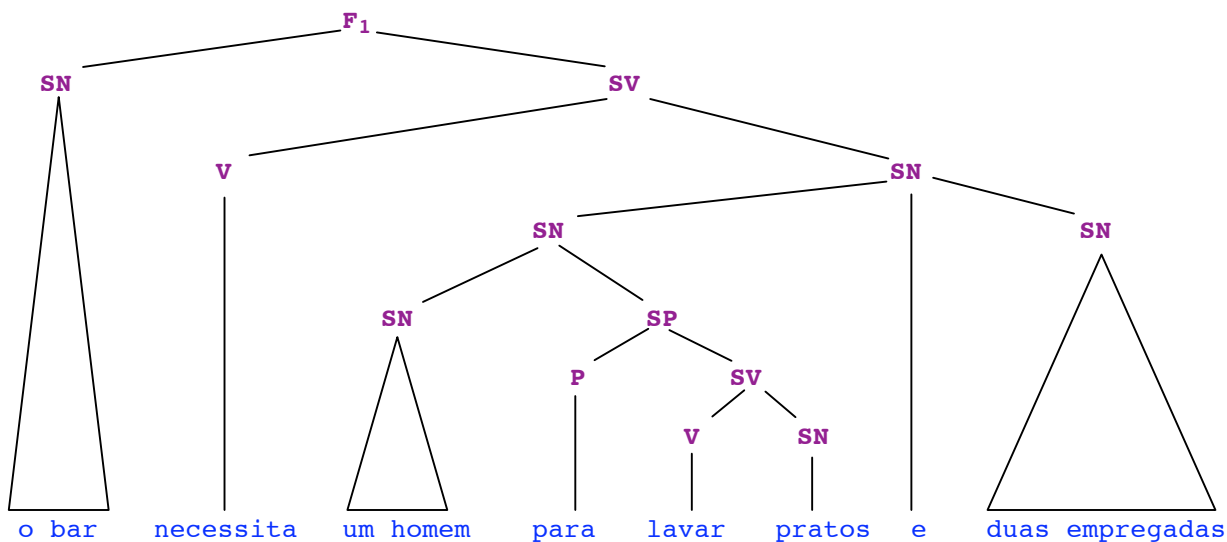
Frase	Correcção sintáctica	Correcção semântica	Correcção pragmática
O aquecimento estão ligado	não	sim	sim
Os pneus são novos	sim	sim	não
Muitas janelas estão a comer o guisado	sim	não	não

Atenção: a avaliação pragmática é dependente da situação que se esteja a considerar. Neste exercício considerou-se que a frase em avaliação foi uma resposta à queixa de que o carro estava muito frio. Também se considerou não se estar em presença de um mundo irreal ou de fantasia!

Problema 2 — Estruturas sintácticas de "As crianças foram encontradas mortas pela mãe"

Algumas considerações sobre as interpretações que se podem associar a cada uma destas estruturas:

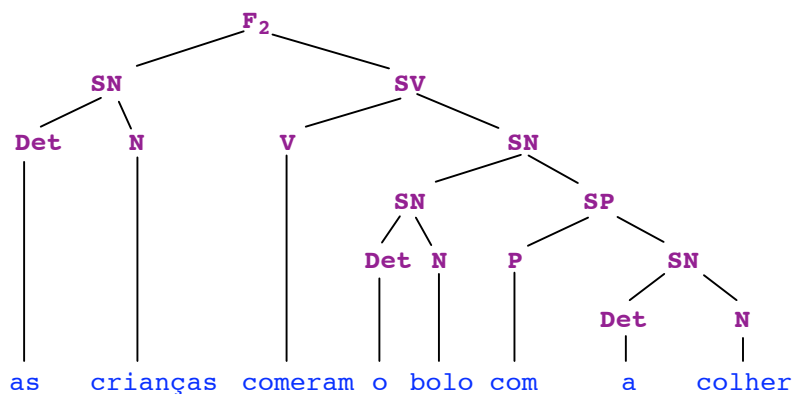
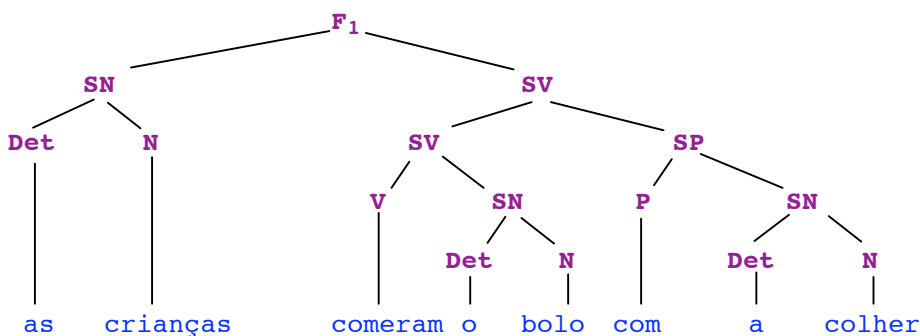
F_1: **crianças:** já estavam mortas quando a mãe as encontrou

F_2: **crianças:** foram mortas pela mãe

Problema 3 — Estruturas sintácticas de "*O bar necessita um homem para lavar pratos e duas empregadas*"

Algumas considerações sobre as interpretações que se podem associar a cada uma destas estruturas:

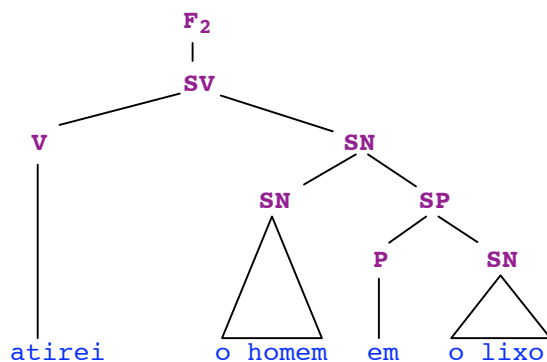
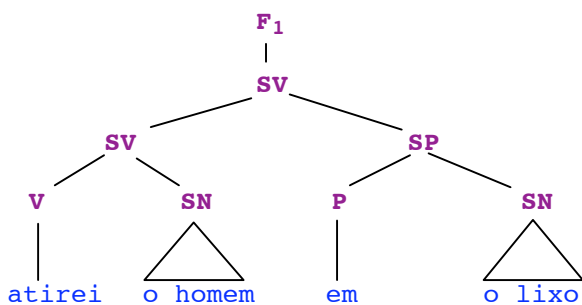
F_1: **bar:** quer contratar três pessoas, um homem e duas mulheres
homem: vai, no bar, lavar pratos

F_2: **bar:** quer contratar uma única pessoa
homem: vai, no bar, lavar pratos e duas mulheres

Problema 4 — Estruturas sintácticas de "*As crianças comeram o bolo com a colher*"

Algumas considerações sobre as interpretações que se podem associar a cada uma destas estruturas:

F_1: **crianças:** comeram o bolo usando uma colher
bolo: ? (nada se sabe sobre o bolo, a não ser que foi comido)

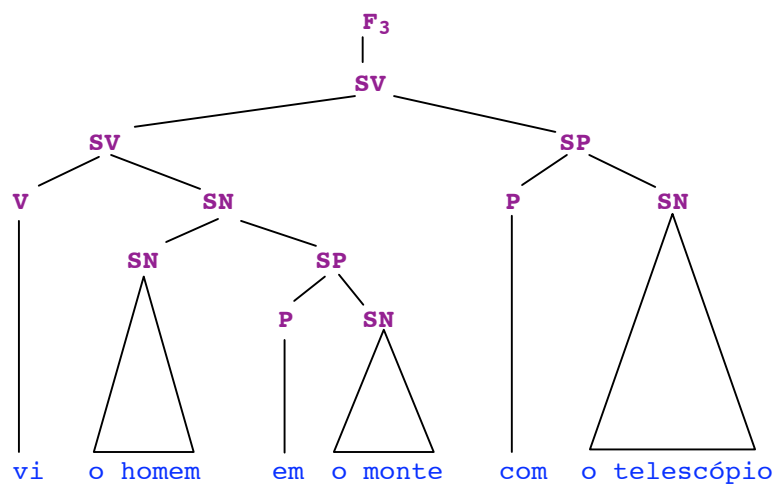
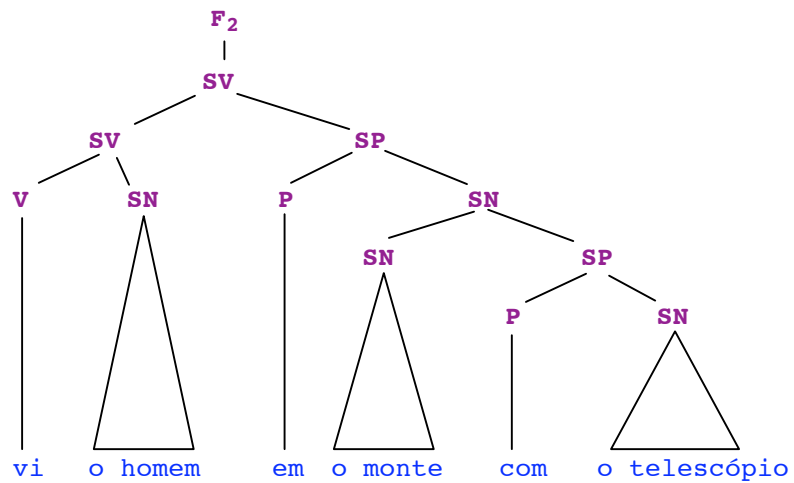
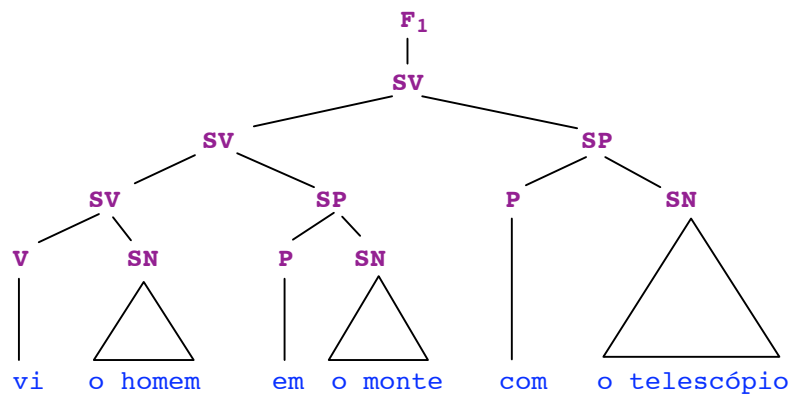
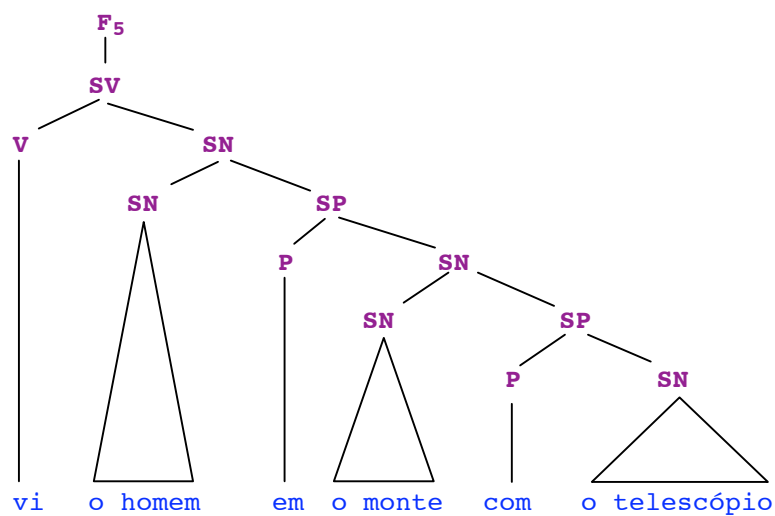
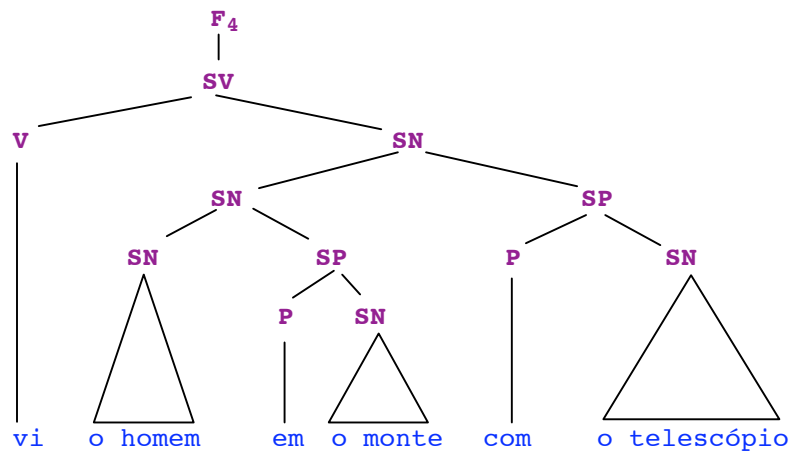
F_2: **crianças:** ? (nada se sabe sobre como as crianças comeram o bolo)
bolo: tem uma colher

Problema 5 — Estruturas sintácticas de "*Atirei o homem em o lixo*"

Algumas considerações sobre as interpretações que se podem associar a cada uma destas estruturas:

F_1: **Eu:** estou no lixo
"o homem": está inicialmente no lixo. O destino é desconhecido

Eu: não estou no lixo
"o homem": vai para dentro do lixo

F_2: **Eu:** tenho muita força, para atirar o homem e o lixo!
"o homem": já está no lixo

Problema 6 — Estruturas sintácticas de "*Vi o homem no monte com o telescópio*"

Algumas considerações sobre as interpretações que se podem associar a cada uma destas estruturas:

F_1: **Eu:** em cima do monte
"o homem":?
"o telescópio": usado para ver o homem

F_2: **Eu:** em cima do monte
"o homem":?
"o telescópio": pertence ao monte

F_3: **Eu:** ?
"o homem":em cima do monte
"o telescópio": usado para ver o homem

F_4: **Eu:** ?
"o homem":em cima do monte
"o telescópio": está com o homem

F_5: **Eu:** ?
"o homem":em cima do monte
"o telescópio": pertence ao monte

Problema 7 — Conversão de "potato" em "batata" com transdutores

É necessário definir os símbolos usados pelos transdutores, ficheiro "letras.syms", e o tradutor, ficheiro "trad.txt" (os símbolos e números estão separados por um "tab"):

Lista de Símbolos [ficheiro letras.syms]

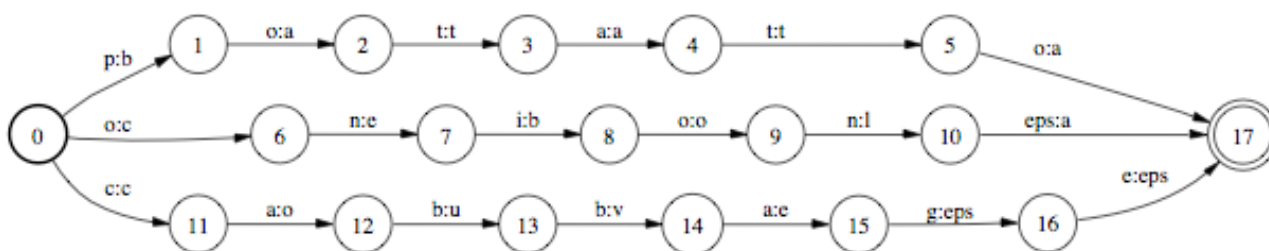
eps	0	n	14
a	1	o	15
b	2	p	16
c	3	q	17
d	4	r	18
e	5	s	19
f	6	t	20
g	7	u	21
h	8	v	22
i	9	w	23
j	10	x	24
k	11	y	25
l	12	z	26
m	13		

Tradutor [ficheiro trad.txt]

0	1	p	b
1	2	o	a
2	3	t	t
3	4	a	a
4	5	t	t
5	6	o	a
0	7	o	c
7	8	n	e
8	9	i	b
9	10	o	o
10	11	n	l
11	6	eps	a
0	12	c	c
12	13	a	o
13	14	b	u
14	15	b	v
15	16	a	e
16	17	g	eps
17	6	e	eps
6			

Para gerar e visualizar o transdutor executam-se os seguintes comandos:

```
fsmcompile -t -i letras.syms -o letras.syms < trad.txt > trad.fsm
fsmdraw -i letras.syms -o letras.syms <  trad.fsm | dot -Tps > trad.ps
```

Contendo o ficheiro "trad.ps" o seguinte grafo:

Para efectuar uma tradução é necessário obter previamente um transdutor com s símbolos que se pretendem traduzir (os símbolos e números estão separados por um "tab"):

potato [ficheiro potato.txt]	**onion** [ficheiro onion.txt]	**cabbage** [ficheiro cabbage.txt]
0 1 p p	0 1 o o	0 1 c c
1 2 o o	1 2 n n	1 2 a a
2 3 t t	2 3 i i	2 3 b b
3 4 a a	3 4 o o	3 4 b b
4 5 t t	4 5 n n	4 5 a a
5 6 o o	5	5 6 g g
6		6 7 e e
		7

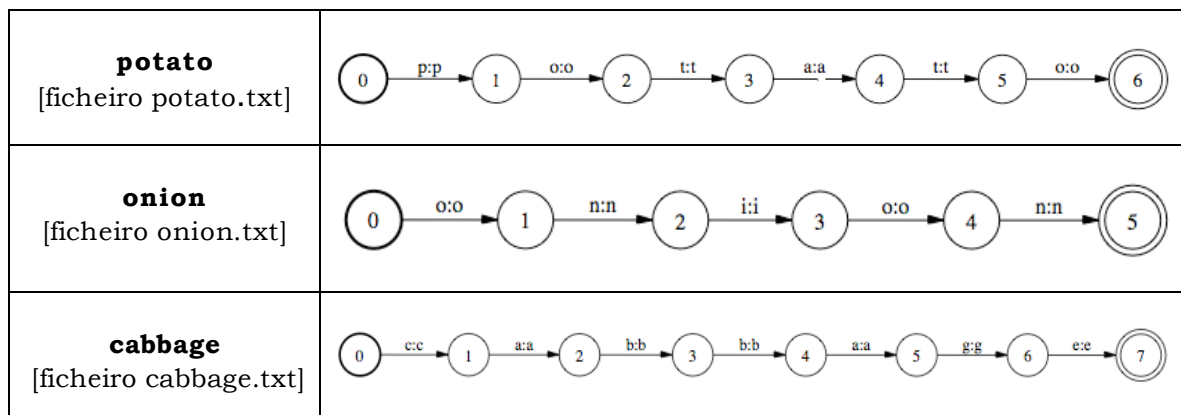
Para gerar e poder visualizar estes três transdutores é necessário executar os seguintes comandos:

```
fsmcompile -t -i letras.syms -o letras.syms < potato.txt  > potato.fsm
fsmdraw  -i letras.syms -o letras.syms <  potato.fsm | dot -Tps > potato.ps

fsmcompile -t -i letras.syms -o letras.syms < cabbage.txt  > cabbage.fsm
fsmdraw  -i letras.syms -o letras.syms <  cabbage.fsm | dot -Tps > cabbage.ps

fsmcompile -t -i letras.syms -o letras.syms < onion.txt  > onion.fsm
fsmdraw  -i letras.syms -o letras.syms <  onion.fsm | dot -Tps > onion.ps
```

Obtendo-se três ficheiros com os transdutores (extensão “fsm”) e três ficheiros que permitem visualizar os transdutores gerados (extensão “ps”):

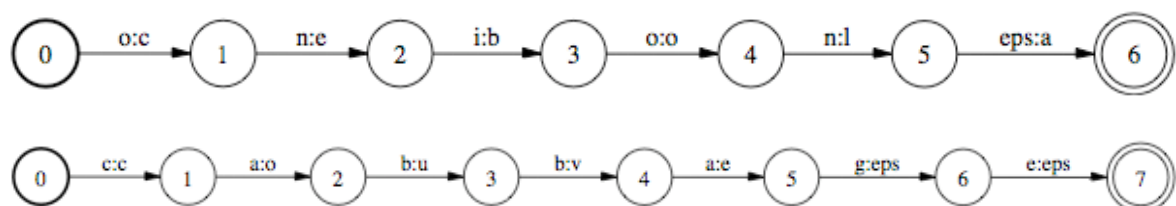
potato [ficheiro potato.txt]	0 –p:p→ 1 –o:o→ 2 –t:t→ 3 –a:a→ 4 –t:t→ 5 –o:o→ 6
onion [ficheiro onion.txt]	0 –o:o→ 1 –n:n→ 2 –i:i→ 3 –o:o→ 4 –n:n→ 5
cabbage [ficheiro cabbage.txt]	0 –c:c→ 1 –a:a→ 2 –b:b→ 3 –b:b→ 4 –a:a→ 5 –g:g→ 6 –e:e→ 7

Para efectuar uma tradução, há que compor o transdutor que descreve a entrada, por exemplo “potato.fsm”, com o transdutor que modela a tradução “trad.fsm”. O programa que permite compor tansdutores é o “fsmcompose”. O próximo comando efectua a composição dos transdutores, obtendo-se um novo transdutor (denominado “batata.fsm”):

```
fsmcompose potato.fsm trad.fsm > batata.fsm
```

Para o visualizar há que executar a seguinte instrução:

```
fsmdraw -i letras.syms -o letras.syms <  batata.fsm | dot -Tps > batata.ps
```

Para traduzir “onion” e “cabbage” é necessário dar os comandos:

```
fsmcompose onion.fsm trad.fsm > cebola.fsm
fsmcompose cabbage.fsm trad.fsm > couve.fsm
fsmdraw -i letras.syms -o letras.syms <  couve.fsm | dot -Tps > couve.ps
fsmdraw -i letras.syms -o letras.syms <  cebola.fsm | dot -Tps > cebola.ps
```

Problema 8 — Dicionário multilingue com transdutores

Para minimizar o esforço, definem-se apenas 3 dicionários e através da operação de inversão e composição é possível obter transdutores que convertam entre quaisquer duas línguas. Os três dicionários têm em comum os símbolos da língua portuguesa que também têm de ser definidos (os símbolos e números estão separados por um “tab”):

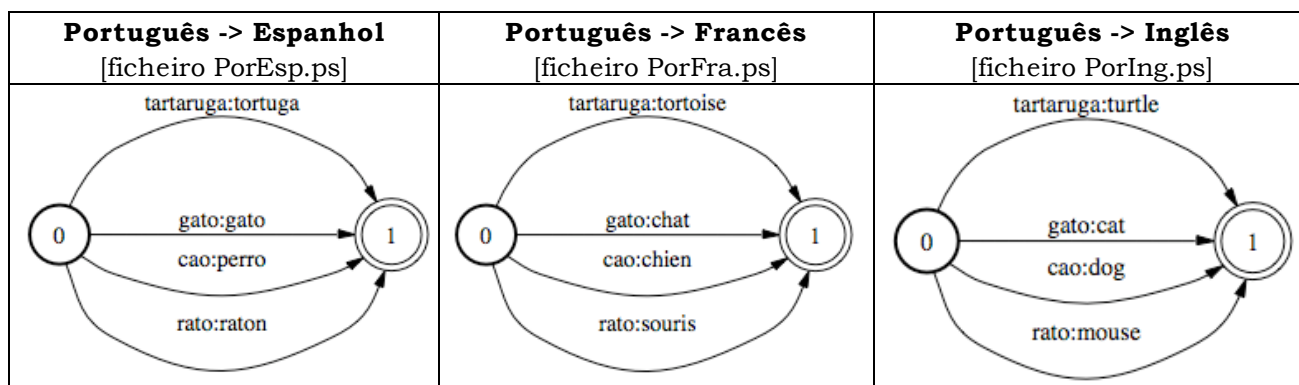
Português -> Espanhol [ficheiro PorEsp.txt]	**Português -> Francês** [ficheiro PorFra.txt]	**Português -> Inglês** [ficheiro PorIng.txt]
0 1 tartaruga tortuga	0 1 tartaruga tortoise	0 1 tartaruga turtle
0 1 gato gato	0 1 gato chat	0 1 gato cat
0 1 cao perro	0 1 cao chien	0 1 cao dog
0 1 rato raton	0 1 rato souris	0 1 rato mouse
1	1	1

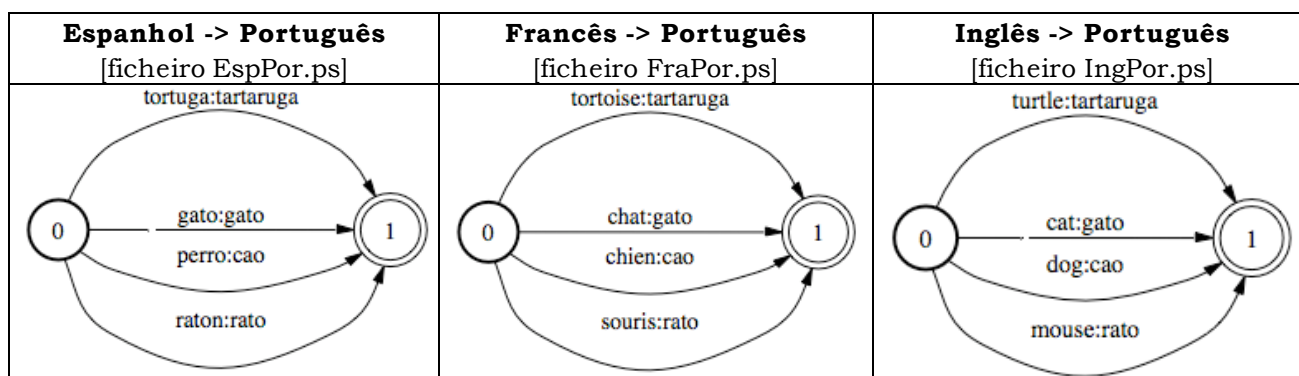
Lista de Símbolos [ficheiro palavras.syms]
eps 0
cao 1
cat 2
chat 3
chien 4
dog 5
gato 6
mouse 7
perro 8
rato 9
raton 10
souris 11
tartaruga 12
tortoise 13
tortuga 14
turtle 15

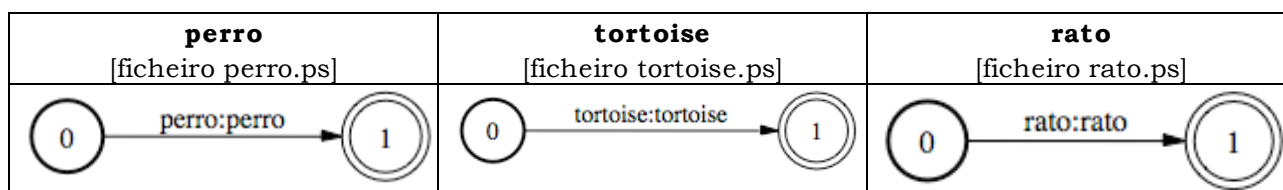
Para gerar os transdutores executam-se os seguintes comandos:

```
fsmcompile -t -i palavras.syms -o palavras.syms < PorEsp.txt > PorEsp.fsm
fsmcompile -t -i palavras.syms -o palavras.syms < PorFra.txt > PorFra.fsm
fsmcompile -t -i palavras.syms -o palavras.syms < PorIng.txt > PorIng.fsm
```

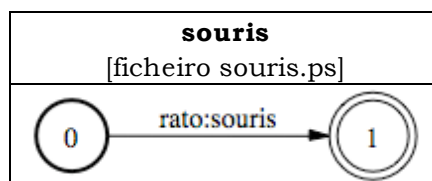
Para visualizar os transdutores gerados é necessário executar as seguintes instruções:

```
fsmdraw -i palavras.syms -o palavras.syms < PorEsp.fsm | dot -Tps > PorEsp.ps
fsmdraw -i palavras.syms -o palavras.syms < PorFra.fsm | dot -Tps > PorFra.ps
fsmdraw -i palavras.syms -o palavras.syms < PorIng.fsm | dot -Tps > PorIng.ps
```

Quando se invertem os transdutores obtidos anteriormente geram-se novos tradutores capazes de traduzir de espanhol, francês e inglês para português:

```
fsminvert PorEsp.fsm > EspPor.fsm
fsminvert PorFra.fsm > FraPor.fsm
fsminvert PorIng.fsm > IngPor.fsm
fsmdraw -i palavras.syms -o palavras.syms < EspPor.fsm | dot -Tps > EspPor.ps
fsmdraw -i palavras.syms -o palavras.syms < FraPor.fsm | dot -Tps > FraPor.ps
fsmdraw -i palavras.syms -o palavras.syms < IngPor.fsm | dot -Tps > IngPor.ps
```

Para efectuar uma tradução é necessário obter um transdutor com o símbolo que se pretende traduzir (os símbolos e números estão separados por um "tab"):

perro [ficheiro perro.txt]	**tortoise** [ficheiro tortoise.txt]	**rato** [ficheiro rato.txt]
`0 1 perro perro` `1`	`0 1 tortoise tortoise` `1`	`0 1 rato rato` `1`

Agora pode-se, por exemplo, compor o transdutor que descreve o símbolo rato com o transdutor que modela a tradução de Português para Francês, obtendo-se um transdutor (ficheiro souris.fsm) que representa o resultado da tradução:

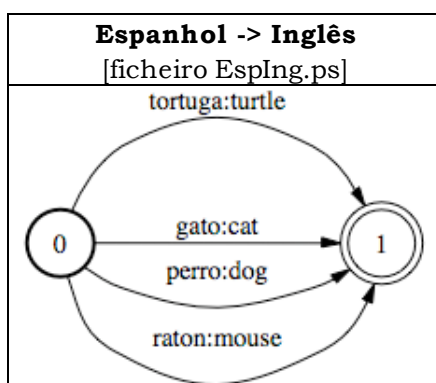
```
fsmcompose rato.fsm PorFra.fsm > souris.fsm
fsmdraw -i palavras.syms -o palavras.syms < souris.fsm | dot -Tps > souris.ps
```

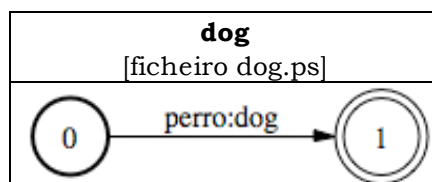
souris
[ficheiro souris.ps]

Mais dois exemplos, um traduzindo “tortoise” de francês para Português e outro traduzindo “perro” de Espanhol para Inglês (composição do transdutor de espanhol para português com o transdutor de português para inglês):

```
fsmcompose tortoise.fsm FraPor.fsm > tartaruga.fsm
fsmdraw -i palavras.syms -o palavras.syms < tartaruga.fsm | dot -Tps > tartaruga.ps
```

tartaruga
[ficheiro tartaruga.ps]

```
fsmcompose EspPor.fsm PorIng.fsm > EspIng.fsm
fsmdraw -i palavras.syms -o palavras.syms < EspIng.fsm | dot -Tps > EspIng.ps
```

Espanhol -> Inglês
[ficheiro EspIng.ps]

```
fsmcompose perro.fsm EspIng.fsm > dog.fsm
fsmdraw -i palavras.syms -o palavras.syms < dog.fsm | dot -Tps > dog.ps
```

dog
[ficheiro dog.ps]

Problema 9 — Distância de Levenshtein entre as palavras “bola” e “rolha”

Para calcular a distância de Levenshtein vai-se usar o algoritmo descrito pela função “número_mínimo_edições_1” que calcula o número mínimo de edições. Para que a distância seja a proposta por Levenshtein é necessário considerar que o custo das inserções, remoções e substituições é sempre 1.

```
função número_mínimo_edições_1(saída, entrada)

 1ª parte (iniciação):
   n ← comprimento(saída)
   m ← comprimento(entrada)
   cria a matriz distância[n+1, m+1]
   distância[0,0] ← 0
   para cada coluna j de 1 a n faz
     distância[j,0] ← j
   para cada linha i de 1 a m faz
     distância[0,i] ← i

 2ª parte (computação):
   para cada linha i de 1 a m faz
     para cada coluna j de 1 a n faz
       distância[j,i] ← min(distância[j-1,i]   + custo_inserção(saída_j),
                            distância[j-1,i-1] + custo_subs(saída_j,entrada_i)
                            distância[j,i-1]   + custo_remoção(entrada_i) )

 3ª parte (devolução do resultado):
   devolve distância[n,m]
```

A aplicação da 1ª parte do algoritmo, correspondente à iniciação da matriz, produz o seguinte resultado:

i	4	**a**	4					
	3	**l**	3					
	2	**o**	2					
	1	**b**	1					
	0	**#**	0	1	2	3	4	5
			#	**r**	**o**	**l**	**h**	**a**
			0	1	2	3	4	5
			j					

O algoritmo vai iniciar o preenchimento das células ainda vazias da matriz, usando a seguinte estratégia: por linhas e da esquerda para a direita. Assim, a primeira célula a preencher é a de coordenadas [1,1]. O valor desta célula é o mínimo de:

i) o custo de partir da cadeia de caracteres de entrada “b” e uma saída vazia “#” (distância[0,1] que tem um custo 1) mais o custo de inserir um “r” na saída (custo 1), ou seja, um custo total de 2;

ii) o custo de partir de uma entrada vazia “#” e de uma saída vazia “#” (distância[0,0] que tem um custo 0) mais o custo de substituir um “b” na entrada por um “r” na saída (custo 1), ou seja, um custo total de 1;

iii) o custo de partir de uma entrada vazia “#” e uma saída que corresponde à cadeia de caracteres “r” (`distância[1,0]` que tem um custo 1) mais o custo de remover um “b” da entrada (custo 1), ou seja, um custo total de 2;

Assim, obtém-se a seguinte matriz:

	4	a	4					
	3	l	3					
i	2	o	2					
	1	b	1	1				
	0	#	0	1	2	3	4	5
			#	r	o	l	h	a
			0	1	2	3	4	5
			i					

Seguindo o mesmo raciocínio, preenche-se a linha correspondente a i=1:

```
distância[2,1] = Mínimo(1+1 ; 1+1 ; 2+1) = 2
distância[3,1] = Mínimo(2+1 ; 2+1 ; 3+1) = 3
distância[4,1] = Mínimo(3+1 ; 3+1 ; 4+1) = 4
distância[5,1] = Mínimo(4+1 ; 4+1 ; 5+1) = 5
```

	4	a	4					
	3	l	3					
i	2	o	2					
	1	b	1	1	2	3	4	5
	0	#	0	1	2	3	4	5
			#	r	o	l	h	a
			0	1	2	3	4	5
			j					

Aplicando o mesmo raciocínio às restantes linhas da matriz, obtém-se:

a	4	4	3	2	2	**2**
l	3	3	2	**1**	**2**	3
o	2	2	**1**	2	3	4
b	1	**1**	2	3	4	5
#	**0**	1	2	3	4	5
	#	r	o	l	h	a

Escolhendo um caminho válido entre as células inicial e final ([0,0] e [5,4]), seleccionam-se implicitamente as operações que transformam a entrada “bola” na saída “rolha”:

a	4	4	3	2	2	**2**
l	3	3	2	**1**	**2**	3
o	2	2	**1**	2	3	4
b	1	**1**	2	3	4	5
#	**0**	1	2	3	4	5
	#	r	o	l	h	a

A distância de Levenshtein é 2 (célula [5,4] da matriz) e corresponde à execução das seguintes operações:

i) a substituição de um "`b`" por um "`r`" (custo 1);
ii) a substituição de um "`o`" por um "`o`" (custo 0);
iii) a substituição de um "`l`" por um "`l`" (custo 0);
iv) a inserção de um "`h`" na saída (custo 1);
v) a substituição de um "`a`" por um "`a`" (custo 0);

Problema 10 — Distância mínima entre “intention” e “execution”

O algoritmo usado neste problema é o expresso pelo pseudo-código da função “número_mínimo_edições_2”:

```
função número_mínimo_edições_2(saída, entrada)
  n ← comprimento(saída)
  m ← comprimento(entrada)
  cria a matriz distância[n+1, m+1]
  distância[0,0] ← 0
  para cada coluna j de 1 a n faz
    distância[j,0] ← distância[j-1,0] + custo_inserção(saída_j)
  para cada linha i de 1 a m faz
    distância[0,i] ← distância[0,i-1] + custo_remoção(entrada_i)
  para cada linha i de 1 a m faz
    para cada coluna j de 1 a n faz
      distância[j,i] ← min( distância[j-1,i]   + custo_inserção(saída_j),
                            distância[j-1,i-1] + custo_subs(saída_j,entrada_i)
                            distância[j,i-1]   + custo_remoção(entrada_i) )
  devolve distância[n,m]
```

A matriz que resulta da aplicação do algoritmo, quando se considera que cada inserção, remoção e substituição tem custo 1, é a seguinte (“intention” é a palavra de origem e “execution” é a palavra de destino):

n	9	8	8	8	8	8	8	7	6	**5**
o	8	7	7	7	7	7	7	6	**5**	6
i	7	6	6	6	6	6	6	**5**	6	7
t	6	5	5	5	5	5	**5**	6	7	8
n	5	4	4	4	4	**5**	6	7	7	7
e	4	3	4	3	**4**	5	6	6	7	8
t	3	3	3	**3**	4	5	5	6	7	8
n	2	2	**2**	3	4	5	6	7	8	7
i	1	**1**	2	3	4	5	6	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Da matriz pode-se concluir que as operações que correspondem ao custo mínimo encontrado são:

i) a substituição de um “i” por um “e” (custo 1);
ii) a substituição de um “n” por um “x” (custo 1);
iii) a substituição de um “t” por um “e” (custo 1);
iv) a substituição de um “e” por um “c” (custo 1);
v) a substituição de um “n” por um “u” (custo 1);
vi) a substituição consecutiva dos caracteres “tion” por “tion” (custo 0);

A matriz que resulta da aplicação do algoritmo, quando se considera que cada inserção e cada remoção tem custo 1 e cada substituição tem custo 2, é a seguinte (“`intention`” é a palavra de origem e “`execution`” é a palavra de destino):

n	9	8	9	10	11	12	11	10	9	**8**
o	8	7	8	9	10	11	10	9	**8**	9
i	7	6	7	8	9	10	9	**8**	9	10
t	6	5	6	7	8	9	**8**	9	10	11
n	5	4	5	6	**7**	**8**	9	10	11	10
e	4	3	4	**5**	6	7	8	9	10	9
t	3	4	**5**	6	7	8	7	8	9	8
n	2	3	**4**	5	6	7	8	7	8	7
i	1	**2**	3	4	5	6	7	6	7	8
#	**0**	**1**	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Da matriz pode-se concluir que as operações (uma das alternativas) que correspondem ao custo mínimo encontrado são:

i) a inserção de um “`e`” (custo 1);
ii) a remoção de um “`i`” (custo 1);
iii) a substituição de um “`n`” por um “`x`” (custo 2);
iv) a remoção de um “`t`” (custo 1);
v) a substituição de um “`e`” por um “`e`” (custo 0);
vi) a substituição de um “`n`” por “`c`” (custo 2);
vii) a inserção de um “`u`” (custo 1);
viii) a substituição consecutiva dos caracteres “`tion`” por “`tion`” (custo 0).

Da notar que agora, existem mais caminhos alternativos por uma substituição ter o mesmo custo que:

i) uma inserção seguida de uma remoção;
ii) uma remoção seguida de uma inserção.

Problema 11 — Cálculo de bigramas num corpus com 300 frases

Recordando a definição de bigramas:

$$P(W_n|W_{n-1}) = \frac{C(W_{n-1}W_n)}{\sum_W C(W_{n-1}W)} = \frac{C(W_{n-1}W_n)}{C(W_{n-1})}$$

conclui-se que a tabela fornecida contém os valores a usar no numerador, e os valores a usar no denominador são o número total de ocorrências de cada uma das categorias.

Por exemplo, para calcular o bigrama P(DET|Ø) (ou seja, a probabilidade de ocorrência de um DET, dado que anteriormente ocorreu um início de frase — Ø):

$$P(DET|\phi) = \frac{C(\phi DET)}{C(\phi)} = \frac{213}{300} = 0.710000000000000$$

Os restantes bigramas apresentam-se de seguida:

Par	Contagem do 1º elemento	Contagem do par	Bigrama
Ø,DET	300	213	P(DET|Ø) = 0.710000000000000
Ø,N	300	87	P(N|Ø) = 0.290000000000000
DET,N	558	558	P(N|DET) = 1.000000000000000
N,V	833	358	P(V|N) = 0.429771908763505
N,N	833	108	P(N|N) = 0.129651860744298
N,P	833	366	P(P|N) = 0.439375750300120
V,N	300	75	P(N|V) = 0.250000000000000
V,DET	300	194	P(DET|V) = 0.646666666666667
P,DET	307	226	P(DET|P) = 0.736156351791531
P,N	307	81	P(N|P) = 0.263843648208469

Problema 12 — Algoritmo de Viterbi em "O rato roeu a rolha"

[Este problema é uma adaptação de um trabalho realizado pelo aluno Sérgio Costa em 2005/06]

O algoritmo de Viterbi, criado por Andrew Viterbi, é um algoritmo de programação dinâmica que permite encontrar a sequência mais provável de estados não observáveis – o chamado "caminho Viterbi" – resultando numa sequência de eventos observáveis, utilizada normalmente no contexto dos modelos de Markov não observáveis.

Actualmente, o algoritmo de Viterbi é frequentemente usado em áreas como o reconhecimento de discurso, detecção de palavras-chave, linguística computacional e bio-informática. Neste exercício, pretende-se estudar a aplicação do algoritmo de Viterbi na resolução do problema da etiquetação morfológica.

O algoritmo não é geral, na medida em que faz uma série de assumpções. Em primeiro lugar, tanto os eventos observáveis como os não observáveis têm de estar em sequência, sendo que estas sequências correspondem, normalmente, a sequências temporais. Em segundo lugar, as duas sequências devem estar alinhadas, e um evento observável tem de corresponder, obrigatoriamente, a um evento não observável. Em terceiro lugar, o cálculo da sequência não observável mais provável num determinado ponto `t`, deve depender apenas do evento observável no ponto `t` e da sequência mais provável no ponto `t−1`. Todas estas assumpções são satisfeitas por um modelo de Markov de primeira ordem.

O algoritmo opera com base em máquinas de estados, existindo um número finito de estados, que pode ser enumerado. Cada estado (ou nó) pode ser alcançado por uma ou mais sequências (ou caminhos). Contudo, existe uma sequência mais provável, o chamado "caminho sobrevivente" (survivor path), que permite alcançar esse estado. Esta é uma das assumpções fundamentais do algoritmo, pois este analisa todos os caminhos que levam a um estado, e escolhe aquele que tem maior probabilidade. Desta forma, o algoritmo não tem de manter o registo de todos os caminhos alternativos, armazenando apenas um caminho por estado. A segunda assumpção chave do algoritmo é o facto de a transição de um estado para um novo estado ser marcada por uma métrica incremental, usualmente um número. Outra das assumpções chaves do algoritmo é o facto de os eventos serem cumulativos ao longo de um caminho, sendo, normalmente, aditivos. Deste modo, o ponto fulcral do algoritmo consiste em associar um número a cada estado. Quando ocorre um evento, o algoritmo passa para um novo conjunto de estados, combinando a métrica de um estado possível anterior com a métrica incremental da transição resultante da ocorrência do evento. Neste passo, o algoritmo escolhe a melhor transição. Note-se que a métrica incremental, associada a um evento, depende da possibilidade de se realizar a transição do estado anterior para o novo estado.

Durante a aplicação do algoritmo, é necessário armazenar um historial da procura. Em alguns casos, o historial da procura é finito, mas pode dar-se o caso de não ser possível armazenar todo o historial e, deste modo, ser necessário limitar a profundidade da procura.

O algoritmo de Viterbi recorre a três estruturas de dados (todas elas vectores):

- `SeqScore` – regista a "pontuação" da melhor sequência encontrada até à posição `t`, com a categoria L_n. O vector tem dimensões [`N` x `n`], em que `N` é o número de categorias léxicas e `n` é o número de palavras na sequência;
- `BackPtr` – regista o estado anterior a um dado estado. O vector tem dimensões [`N` x `n`];
- `C` – regista o resultado, isto é, a melhor sequência de etiquetas. O vector tem dimensões [`1` x `n`].

O algoritmo pode ser divido em três etapas: Iniciação, Iteração e Identificação da Sequência. Assim, dada a sequência de palavras $w_1, \ldots, w_n$, as categorias léxicas $L_1, \ldots, L_N$, encontra-se a sequência mais provável de categorias léxicas $C = [C_1, \ldots, C_n]$ do seguinte modo:

```
Iniciação
Para i=1 até N faz
    SeqScore[i,1] = P(w1|Li)*P(Li|Ø)
    BackPtr[i,1] = 0
Iteração
Para t=2 até n faz
    Para i=1 até N faz
        SeqScore[i,t] = MAXj=1,N(SeqScore[j,t-1]*P(Li|Lj))*P(wt|Li)
        BackPtr[i,t] = índice de j que resultou na pontuação máxima
Identificação da Sequência
    C[n] = I que maximiza SeqScore[i,n]
    Para i=n-1 até 1 faz
        C[i] = BackPtr[C(i+1),i+1]
```

No pseudo-código representado acima:

- Ø representa uma categoria léxica fictícia, utilizada apenas para o início da frase;
- Na etapa de iniciação, a primeira coluna do vector BackPtr toma o valor zero em todas as linhas, pois nenhuma palavra precede a primeira palavra;
- Na etapa final, de identificação da sequência de etiquetas, encontra-se, na última coluna do vector SeqScore, a entrada com o valor mais alto. O índice dessa entrada indica a categoria léxica à qual pertence a última palavra;
- As categorias léxicas das palavras anteriores são encontradas fazendo retrocesso com o vector BackPtr, isto é, a entrada do vector o vector BackPtr[C(i+1),i+1] indica a categoria léxica da palavra i.

As categorias léxicas têm a seguinte codificação: L_1=N, L_2=V, L_3=DET, L_4=P.

A execução do algoritmo começa na etapa de iniciação, pelo que é necessário calcular as probabilidades de geração léxica (lexical-generation probabilities) que são estimadas através da contagem do número de ocorrências de cada palavra, por categoria (dados fornecidos no enunciado):

P(o|DET)	250/558 = 4.48028673835E-01
P(o|P)	87/307 = 2.83387622150E-01
P(rato|N)	28/833 = 3.36134453782E-02
P(roeu|V)	1/300 = 3.33333333333E-03
P(a|DET)	208/558 = 3.72759856631E-01
P(a|P)	54/307 = 1.75895765472E-01
P(rolha|N)	5/833 = 6.00240096038E-03

P(DET|Ø)	0.710000000000000
P(N|Ø)	0.290000000000000
P(N|DET)	1.000000000000000
P(V|N)	0.429771908763505
P(N|N)	0.129651860744298
P(P|N)	0.439375750300120
P(N|V)	0.250000000000000
P(DET|V)	0.646666666666667
P(DET|P)	0.736156351791531
P(N|P)	0.263843648208469

A tabela mais à direita resume os resultados referentes ao corpus de 300 frases. Assume-se que qualquer bigrama não listado tem probabilidade 1.0E-06.

Agora já se pode preencher a primeira coluna dos vectores SeqScore e BackPtr:

```
SeqScore[1,1]=P(o|N)*P(N|Ø)= 0.000001*0.29 = 2.9E-07
BackPtr[1,1]=0
SeqScore[2,1]=P(o|V)*P(V|Ø)= 0.000001*0.000001 = 1.0E-12
BackPtr[2,1]=0
SeqScore[3,1]=P(o|DET)*P(DET|Ø)= 0.425170068027*0.71= 3.18100358423E-01
BackPtr[3,1]=0
SeqScore[4,1]=P(o|P)*P(P|Ø)=0.28338762214983*0.000001= 2.83387622150E-07
BackPtr[4,1]=0
```

Apresenta-se em seguida, na forma de uma tabela, o cálculo realizado na etapa de iteração:

t	i	SeqScore[i,t]	BackPtr[i,t]
2	1	MAX(SeqScore[1,1]*P(N|N), SeqScore[2,1]*P(N|V), SeqScore[3,1]*P(N|DET), SeqScore[4,1]*P(N|P)) *P(rato|N) = 1.06924490226E-02	3
2	2	MAX(SeqScore[1,1]*P(V|N), SeqScore[2,1]*P(V|V), SeqScore[3,1]*P(V|DET), SeqScore[4,1]*P(V|P)) *P(rato|V) = 3.18100358423E-13	3
2	3	MAX(SeqScore[1,1]*P(DET|N), SeqScore[2,1]*P(DET|V), SeqScore[3,1]*P(DET|DET),SeqScore[4,1]*P(DET|P)) *P(rato|DET) = 3.18100358423E-13	3
2	4	MAX(SeqScore[1,1]*P(P|N), SeqScore[2,1]*P(P|V), SeqScore[3,1]*P(P|DET), SeqScore[4,1]*P(P|P)) *P(rato|P) = 3.18100358423E-13	3
3	1	MAX(SeqScore[1,2]*P(N|N), SeqScore[2,2]*P(N|V), SeqScore[3,2]*P(N|DET), SeqScore[4,2]*P(N|P)) *P(roeu|N) = 1.38629591170E-09	1
3	2	MAX(SeqScore[1,2]*P(V|N), SeqScore[2,2]*P(V|V), SeqScore[3,2]*P(V|DET), SeqScore[4,2]*P(V|P)) *P(roeu|V) = 1.53177140860E-05	1
3	3	MAX(SeqScore[1,2]*P(DET|N), SeqScore[2,2]*P(DET|V), SeqScore[3,2]*P(DET|DET), SeqScore[4,2]*P(DET|P)) *P(roeu|DET) = 1.06924490226E-14	1
3	4	MAX(SeqScore[1,2]*P(P|N), SeqScore[2,2]*P(P|V), SeqScore[3,2]*P(P|DET), SeqScore[4,2]*P(P|P)) *P(roeu|P) = 4.69800281186E-09	1
4	1	MAX(SeqScore[1,3]*P(N|N), SeqScore[2,3]*P(N|V), SeqScore[3,3]*P(N|DET), SeqScore[4,3]*P(N|P)) *P(a|N) = 3.82942852151E-12	2
4	2	MAX(SeqScore[1,3]*P(V|N), SeqScore[2,3]*P(V|V), SeqScore[3,3]*P(V|DET), SeqScore[4,3]*P(V|P)) *P(a|V) = 5.95791040081E-16	1
4	3	MAX(SeqScore[1,3]*P(DET|N), SeqScore[2,3]*P(DET|V), SeqScore[3,3]*P(DET|DET), SeqScore[4,3]*P(DET|P)) *P(a|DET) = 3.69235602628E-06	2
4	4	MAX(SeqScore[1,3]*P(P|N), SeqScore[2,3]*P(P|V), SeqScore[3,3]*P(P|DET), SeqScore[4,3]*P(P|P)) *P(a|P) = 1.07138956164E-10	1
5	1	MAX(SeqScore[1,4]*P(N|N), SeqScore[2,4]*P(N|V), SeqScore[3,4]*P(N|DET), SeqScore[4,4]*P(N|P)) *P(rolha|N) = 2.21630013582E-08	3
5	2	MAX(SeqScore[1,4]*P(V|N), SeqScore[2,4]*P(V|V), SeqScore[3,4]*P(V|DET), SeqScore[4,4]*P(V|P)) *P(rolha|V) = 3.69235602628E-18	3
5	3	MAX(SeqScore[1,4]*P(DET|N), SeqScore[2,4]*P(DET|V), SeqScore[3,4]*P(DET|DET), SeqScore[4,4]*P(DET|P)) *P(rolha|DET) = 7.88710231044E-17	4
5	4	MAX(SeqScore[1,4]*P(P|N), SeqScore[2,4]*P(P|V), SeqScore[3,4]*P(P|DET), SeqScore[4,4]*P(P|P)) *P(rolha|P) = 3.69235602628E-18	3

Deste modo, o conteúdo final dos vectores SeqScore e BackPtr são, respectivamente:

SeqScore		n (palavras)				
		1 (O)	3 (rato)	3 (roeu)	4 (a)	5 (rolha)
N (categorias)	1 (N)	2.9E-07	1.1E-02	1.4E-09	3.9E-12	2.2E-08
	2 (V)	1.0E-12	3.2E-13	1.5E-05	6.0E-16	3.7E-18
	3 (DET)	3.2E-01	3.2E-13	1.1E-14	3.7E-06	7.9E-17
	4 (P)	2.8E-07	3.2E-13	4.7E-09	1.1E-10	3.7E-18

BackPtr		n (palavras)				
		1 (O)	2 (rato)	3 (roeu)	4 (a)	5 (rolha)
N (categorias)	1 (N)	0	3	1	2	3
	2 (V)	0	3	1	1	3
	3 (DET)	0	3	1	2	4
	4 (P)	0	3	1	1	3

Terminada a etapa de iteração, falta apenas construir a sequência de etiquetas mais provável. Começamos por notar que o valor mais alto para a última coluna de SeqScore encontra-se na primeira linha. Deste modo, C(n)=C(5)=1, isto é, a etiqueta da última palavra (rolha) é N (nome). As restantes etiquetas calculam-se do seguinte modo:

```
C[4] = backPtr[C[5],5] = backPtr[1,5] = 3
C[3] = backPtr[C[4],4] = backPtr[3,4] = 2
C[2] = backPtr[C[3],3] = backPtr[2,3] = 1
C[1] = backPtr[C[2],2] = backPtr[1,2] = 3
```

Ou seja, o conteúdo final do vector C é:

C	n (palavras)				
	1 (O)	2 (rato)	3 (roeu)	4 (a)	5 (rolha)
	3	1	2	3	1

Assim C=[3,1,2,3,1] o que equivale a dizer que o foi etiquetado como DET (determinante), rato foi etiquetado como N (nome), roeu foi etiquetado como V (verbo), a foi etiquetado como DET (determinante) e rolha foi etiquetado como N (nome).

Em termos gráficos, o resultado da *etapa de iteração* é o seguinte (as setas mais carregadas representam a sequência mais provável, o resultado da *identificação da sequência*).